Bluetooth and WLAN Coexistence: Challenges and Solutions

N. Golmie, N. Chevrollier, and O. Rebala
National Institute of Standards and Technology
Gaithersburg, Maryland 20899
Email: nada.golmie@nist.gov

Abstract

In this article we discuss solutions to the interference problem caused by the proximity and simultaneous operation of Bluetooth and WLAN networks. We consider different techniques that attempt to avoid time and frequency collisions of WLAN and Bluetooth transmissions. We conduct a comparative analysis of their respective performance and discuss the trends and trade-offs they bring for different applications and interference levels. Performance is measured in terms of packet loss, TCP goodput, delay, and delay jitter.

I. Introduction

The Bluetooth technology [1] is considered a Wireless Personal Area Network (WPAN) system, intended for cable replacement and short distance ad hoc connectivity. WPAN is distinguished from other types of wireless networks in both size and scope. Communications in WPAN are normally confined to a person or object and extend up to 10 meters in all directions. This is in contrast to Wireless Local Area Networks (WLANs) employing the IEEE 802.11 specifications [2] that typically cover a moderately sized geographic area such as a single building or campus. In this sequel, we will use WLAN and IEEE 802.11 interchangeably. WLANs operate in the 100 meter range and are intended to augment rather than replace traditional wired LANs. They are often used to provide the final few feet of connectivity between the main network and the user.

However, instead of competing with WLANs for applications, WPANs are intended to augment many of the usage scenarios and operate in conjunction with WLANs, i.e., come together in the same laptop, or operate in proximity in an office or conference room environment. For example, Bluetooth can be used to connect a headset, or PDA to a desktop computer, that in turn may be using WLAN to connect to an Access Point placed several meters away.

Bluetooth and several cordless phone manufacturers plan to operate in the 2.4 GHz Industry Scientific and Medical (ISM) unlicensed band since it is suitable for low cost radio solutions such as the ones proposed for WPANs. In addition, IEEE 802.11 [2] has standards for WLANs operating in this band as well. However, the major down side of the unlicensed ISM band is that frequencies must be shared and potential interference tolerated as defined in the the Federal Communications Commission Title 47 of the Code for Federal Regulations Part 15 [3]. While the spread spectrum and power rules are fairly effective in dealing with multiple users in the band provided the radios are physically separated, the same is not true for close proximity radios such as IEEE 802.11 and Bluetooth that may likely come together in a laptop or a desktop. An issue of growing interest is the coexistence of these devices in the same environment.

Recently, there has been a growing number of industry led activities focused on the coexistence of wireless devices in the 2.4 GHz band. Both, the IEEE 802.15.2 Coexistence Task Group [4] and the Bluetooth Special Interest Group (SIG) are looking at similar techniques for alleviating the impact of interference. The proposals considered by the groups are intended for Bluetooth and IEEE 802.11 direct sequence spread spectrum protocols. They range from collaborative schemes to be implemented in the same device to fully independent solutions that rely on interference detection and estimation. Except for a Time Division Multiple Access (TDMA) technique aimed at time sharing the Bluetooth and 802.11 signals [5], most mechanisms considered do not require any direct communication between the protocols. These so-called non-collaborative mechanisms are intended mainly for Bluetooth since it is easier for a frequency hopping system to avoid frequencies occupied by a spread spectrum system such as WLAN. The techniques considered range from adaptive frequency hopping [6] to packet scheduling and traffic control [7]. The techniques used for detecting the presence of WLAN devices in the band are based on measuring the bit or frame error rate, the signal strength or the signal to interference ratio (often implemented as the Received Signal Strength Indicator (RSSI)). For example, each device can maintain a packet error rate measurement per frequency visited. Frequency hopping devices can then know which frequencies are occupied by other users of the band and modify their frequency hopping pattern. They can even choose not to transmit on a certain frequency if that frequency is occupied. The first technique is known as adaptive frequency hopping, while the second technique is known as Medium Access Control (MAC) scheduling. Other scheduling techniques known as packet encapsulation rules or OverLap Avoidance (OLA) [8], use the variety of Bluetooth packet lengths to avoid the overlap in frequency between 802.11 and Bluetooth. In other words, the Bluetooth scheduler knows to use the packet length of proper duration (1, 3 or 5 slots) in order to skip the so-called ”bad” frequency. This was shown to provide goodput improvements for both 802.11 and Bluetooth data traffic.

In this article, we investigate two solutions to the interference problem, namely, (1) an Adaptive Frequency Hopping (AFH) mechanism aimed at modifying the Bluetooth frequency hopping sequence in the presence of WLAN direct sequence spread spectrum devices [9], (2) a Bluetooth Interference Aware Scheduling (BIAS) strategy that postpones the transmission of packets on so-called “bad” frequencies [7]. Each of these two techniques considered imposes a number of implementation implications.

For example, the implication with AFH is that the chipset has to be modified in order to support a new Bluetooth hopping sequence that does not contain any frequencies used by WLAN. On the other hand, the backoff strategy applies to the Bluetooth master device firmware that is responsible for transmitting packets on the medium.

The remainder of this article is organized as follows. Section II discusses interference detection methods used to determine the presence of WLAN interference. In section III and IV, we describe the backoff and AFH procedures respectively. In section V, we consider realistic scenarios to discuss performance trends and trade-offs. In section VI, we offer some concluding remarks.

II. Bluetooth Interference Estimation

Central to most interference mitigation techniques is the ability to detect the presence of other systems operating in the band, or in other words, estimate interference. Techniques that do not require interference estimation belong to the collaborative category where both the Bluetooth and WLAN protocols are implemented on the same device in order for each protocol to be aware of the traffic and packet transmissions in both the WLAN and the Bluetooth networks.

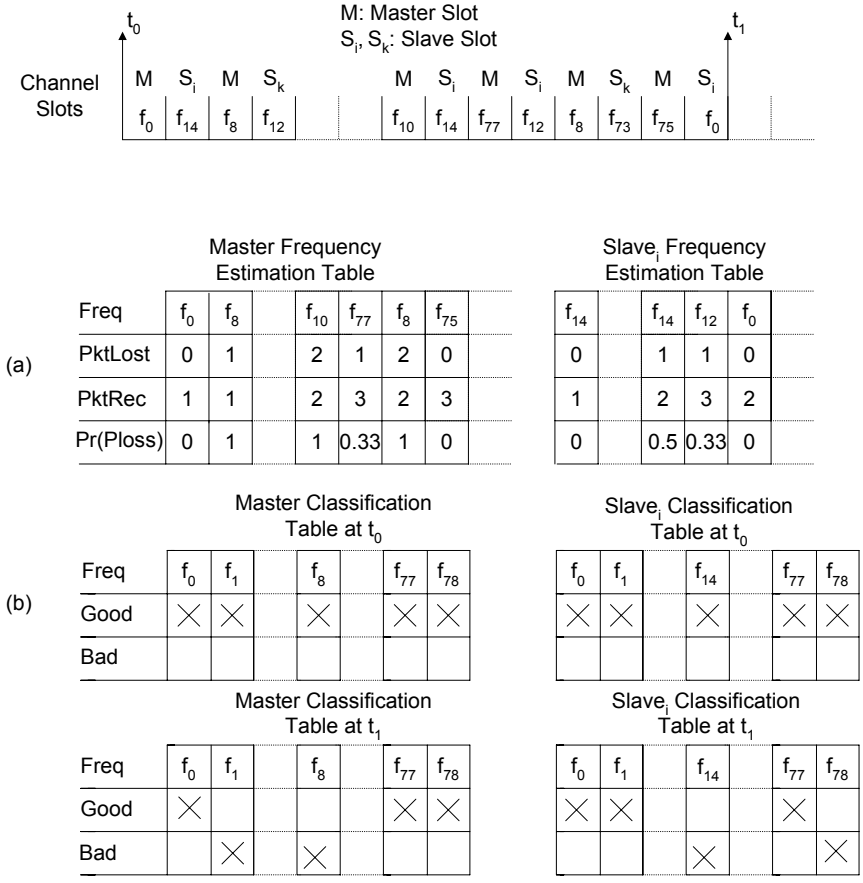

Fig. 1. Interference Estimation and Frequency Classification

Interference estimation methods include Signal to Interference Ratios (SIR), Bit Error Rate (BER) calculation, packet loss, or frame error rate measurements performed by a device receiver. We use packet loss measurements in our performance evaluation although other measurements can be used as well without affecting the outcome of the experiments studied. In addition, we limit our discussion to interference estimation for Bluetooth since that pertains to the solutions presented here.

In a nutshell, here is how a Bluetooth receiver detects the presence of a WLAN spread spectrum system. Measurements are collected by each receiver in the piconet since interference depends on the device location and transmitted power. These measurements consist of a percentage of packets dropped due to errors, Pr(Ploss), that is associated with each frequency in the hopset, f, as shown in Figure 1(a). Given Pr(Ploss) and a packet loss threshold, frequencies are classified "good" or "bad" depending on whether their packet loss rate is less than or greater than the threshold value respectively. In Figure 1(b), we use a packet loss threshold equals to 0.5.

Since in a Bluetooth piconet, the master device controls all packet transmissions, the measurements collected by the slaves are mostly useful if available at the master. There are at least two ways of sharing these measurements among the devices of the piconet. One approach would be for the master and slaves to periodically exchange their measurements via management messages.

Another method would be for the master to derive information about each slave's measurements by looking at the ACK bit sent in the slave's response packets. Observe that in this latter approach, the master can make use of the ACK feedback information as soon as it becomes available, and thus speed up the estimation time by few tens to hundreds of milliseconds depending on the traffic load and packet sizes considered. Scanning the entire frequency band using ACK feedback may take between 0.5 to 1.5 seconds depending on the application and the traffic load considered.

A final point of observation is concerned with the classification update interval. Since the master uses the packet loss information collected in order to rearrange the frequency hopping pattern in case of AFH and/or selectively avoid packet transmissions on so-called "bad" frequencies, one needs to ask how often should frequencies be classified? If the classification update period is relatively short, the classification reflects more accurately the state of the channel at a higher communication overhead cost in case the measurements are distributed via management messages. Also, frequent classifications may lead to a higher packet loss. On the other hand, a long classification period may not be able to keep up with rapid changes in the interference environment, when traffic is bursty and users are mobile. A number of techniques can be used in order to make the update interval track changes in the channel dynamics. In our evaluation, we fixed the update interval to 4 seconds in order to highlight the effects of synchronization messages.

III. Bluetooth Interference Aware Scheduling

Since the interference mitigation approach that we discuss is concerned primarily with packet scheduling and transmission in Bluetooth, we will first give a brief overview of how packets are transmitted in Bluetooth, and we will then show how to modify the packet scheduler in order to mitigate interference.

The Bluetooth transmission channel is divided into 625 μs slots. Transmission occurs in packets that occupy an odd number of slots (1, 3, or 5). Each packet is transmitted on a different hop frequency with a maximum frequency hopping rate of 1600 hops/s in case packets occupy a single slot, and a minimum hopping rate of 320 hops/s in case packets occupy 5 slots. Note that every slot has a frequency associated with it; however transmission of a packet occupying multiple slots always uses the frequency associated with the first slot.

A slave packet always follows a master packet transmission as illustrated in Figure 2(a), which depicts the master's view of the slotted channel. A slave needs to respond to a master's packet that is specifically addressed to it. In case it does not have any data to send, it sends a NULL packet. Moreover, each packet contains the ACK information of the previous packet received.

Since the master is in charge of all transmissions in the piconet and chooses which slave to transmit to, it is easy to envision a scheduling policy at the master that considers the frequency classification information before sending packets on the medium. The so-called Bluetooth Interference Aware Scheduling (BIAS) [7] is a backoff policy that postpones the transmission of a packet until a slot associated with a "good" frequency becomes available. Here is how it works. The master continuously classifies each frequency as either "bad" or "good" based on a predefined criterion, for example a packet loss threshold as mentioned in section II. Given a master/slave slot pair and their associated frequencies as illustrated in Figure 2(a), the master transmits in a slot after it verifies that both the slave's receiving frequency and its own receiving frequency are "good". Thus, the master avoids receiving data on a "bad" frequency, by avoiding a transmission on a frequency preceding a "bad" one in the hopping pattern. If either frequency in the pair is "bad", the master skips the current transmission slot and repeats the procedure over again in the next transmission opportunity.

Finally, Figure 2(a) shows an example of transmission priority that can be built into the master scheduler. In this case, the master schedules retransmissions first, then data packet, and finally acknowledgment packets. Note that in all three cases the pair of frequencies associated with the master and slave slots need to be "good". Additional considerations including bandwidth requirements and quality of service guarantees for each master/slave connection in the piconet can also be combined with the channel state information and mapped into transmission priorities given to each direction in the master/slave communication. Details on assigning transmission priorities are given in [7].

IV. Bluetooth Adaptive Frequency Hopping

The key idea in BIAS is to wait for a slot associated with a "good" frequency in order to transmit a packet. The question that comes up is, can the frequency and slot association be modified in order to eliminate the so-called "bad" frequencies? In other words, can "bad" frequencies be replaced with "good" ones so that transmissions need not be postponed? That's the main idea in adaptive frequency hopping.

First, we describe the Bluetooth frequency hopping sequence defined in the Bluetooth specifications [1] , then we present an AFH algorithm that modifies it in order to mitigate interference.

Frequency hopping in Bluetooth is achieved as follows. Frequencies are sorted into a list of even and odd frequencies in the 2.402-2.480 GHz range. A segment consisting of the first 32 frequencies in the sorted list is chosen. After all 32 frequencies in that window are visited once in a random order, a new window is set including 16 frequencies of the previous window and 16 new frequencies in the sorted list. From the many AFH algorithms possible, here is an implementation that eliminates "bad" frequencies in the sequence.

Given a segment of 32 "good" and "bad" frequencies, the algorithm visits each "good" frequency exactly once. Each "bad" frequency in the segment is replaced with a "good" frequency selected from outside the original segment of 32 as shown in

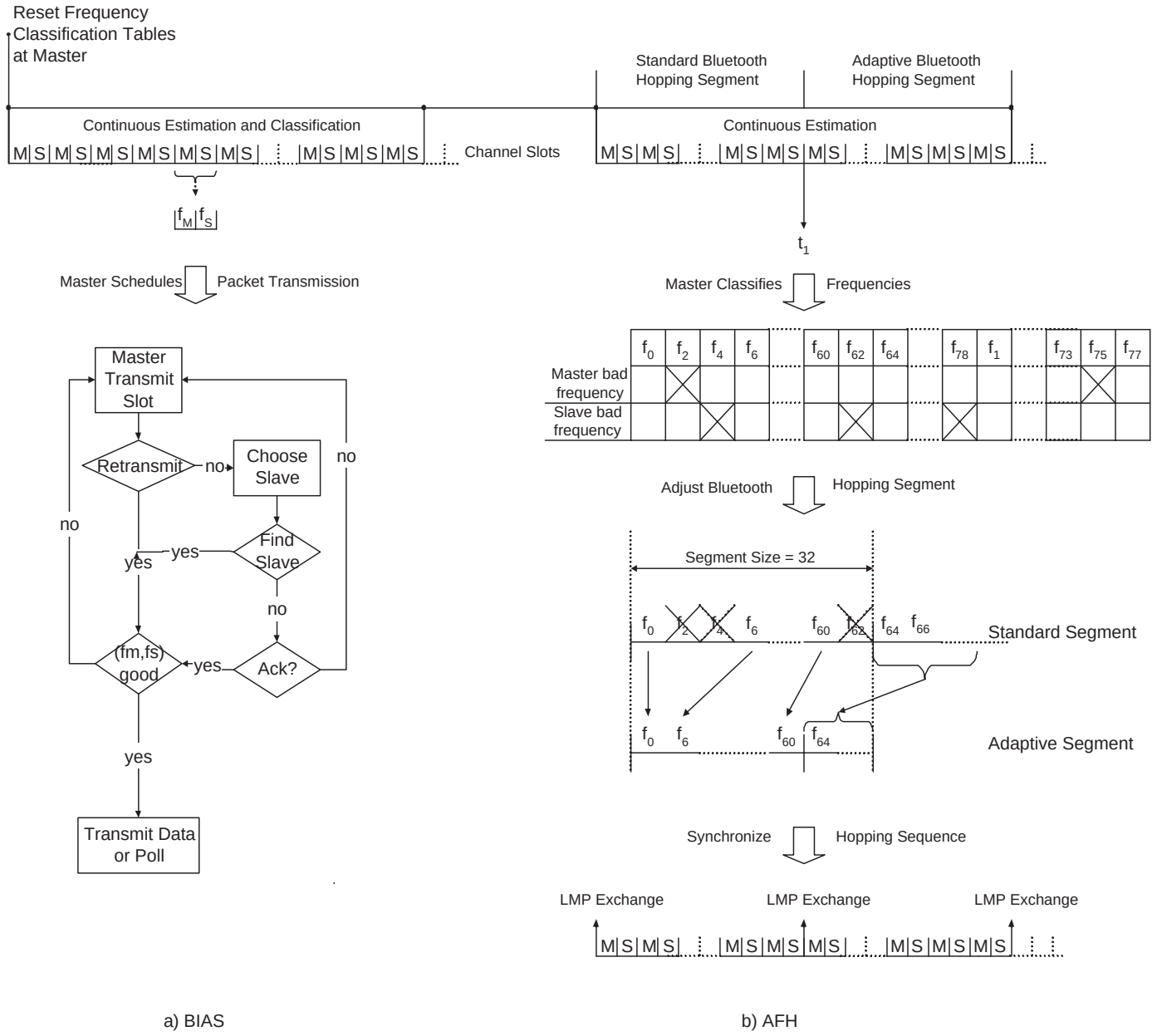

Fig. 2. Bluetooth Scheduling and Adaptive Hopping Techniques

Figure 2(b). Thus, the difference between AFH and the original Bluetooth hopping sequence algorithm is in the selection of only "good" frequencies in order to fill up the segment size. Some additional constraints can be imposed on the maximum number of "bad" frequencies to eliminate if a minimum number of different frequencies is to be kept in the sequence. In their most recent ruling the FCC recommends using at least 15 different frequencies.

Changing the frequency patterns requires changes in the Bluetooth hardware implementations. Another requirement is the advertisement of the new hopping pattern among devices in the piconet in order to keep synchronization. This is typically done using Link Management Protocol (LMP) messages exchanged between the master and the slaves in the piconet in order to advertise the new hopping sequence. This last requirement imposes some limitations on how often a new hopping pattern should be advertised and used. Improving performance such as lowering the packet loss, the access delay, and increasing the throughput should outweigh the communication overhead associated with synchronization. As suggested in section II, the synchronization update interval could be dynamically adjusted so that it tracks changes in the channel. In our simulations the LMP messages were sent twice in a 4 seconds update interval. The first LMP message was sent when the frequency tables were reset, while the second message was sent about 1.5 seconds later to signify the use of a new hopping pattern.

Finally, AFH does not preclude additional scheduling techniques to control the transmission (and possibly the retransmission) of packets on the medium.

V. Performance Evaluation Results

We present simulation results to evaluate the performance of Bluetooth and WLAN and discuss some of the trade-offs associated with the backoff and the frequency hopping schemes presented earlier. Our simulation environment is based on detailed MAC, PHY and channel models for Bluetooth and IEEE 802.11 (WLAN) as described in [10]. The channel model consists of a geometry-based propagation model for the signals, as well as a noise model based on Additive white Gaussian noise (AWGN). For the indoor channel, we apply a propagation model consisting of two parts: (1) line-of-sight propagation (free-space) for the first 8 meters, and (2) a propagation exponent of 3.3 for distances over 8 meters [11]. The transmitters, channel, and receivers are implemented at complex baseband. We develop models for the Bluetooth and the IEEE 802.11 access protocols using the OPNET network simulator and configure the applications available in the simulator library.

In general, we find that performance results vary according to the network configuration, usage scenario and application considered [10]. In this paper, we vary the application and the interference level considered, as these two factors are most likely to dominate the performance results.

For Bluetooth, we consider two applications, FTP and voice. FTP is a bandwidth hungry application that stresses the throughput requirement, while voice has strict delay and jitter requirements. Together, these two applications constitute a representative set of the application space used in a Bluetooth piconet. For WLAN, we use FTP to upload a large file (for instance a movie) to a server. For the FTP profile, the parameters are the inter-request time and the file size. The inter-request time is the interval between two FTP commands, and the file size represents the size of the file requested in bytes. For Bluetooth we vary the file sizes from 200 bytes to 500 Kbytes (every 5 seconds), while for WLAN we use a single file of 960 Mbytes. The voice application used in Bluetooth is based on the G.723.1 encoder (with silence). The simulation and profile parameters are given in Table I.

TABLE I
Simulation Parameters

Simulation Parameters		**Values**
Propagation delay		5 μs/km
Length of simulation run		1600 seconds
Bluetooth Parameters		
ACL Baseband Packet Encapsulation		DH5
Transmitted Power		1 mW
WLAN Parameters		
Transmitted Power		25 mW
Packet Header		224 bits
Packet Payload		12,000 bits
Application Profile Parameters	**Distribution**	**Values**
Bluetooth FTP		
Inter-Request Time (seconds)	Exponential	5
File Size (Kbytes)	varies in	[0.2,500]
WLAN FTP		
File Size (Mbytes)	Constant	960
Bluetooth Voice		
Encoder		G.723.1
Silence Length (seconds)	Exponential	0.65
Talk Spurt (seconds)	Exponential	0.352

We use the four-device configuration shown in Figure 3 that is common to some office or home environments. It consists of a laptop computer connected to the Internet via WLAN, while a desktop located at a distance d from it, is also connected to either a PDA or a wireless headset over a Bluetooth link. By varying d, the level of interference on each of the Bluetooth and WLAN receivers is effected. For example, as d is increased, the level of interference is decreased. Other usage scenarios can also be obtained by putting both WLAN and Bluetooth receivers on the same device, for example the laptop computer in this case. Although some variations in the performance results are to be expected, the differences in the results remain minor.

Now, we discuss the details of two experiments involving a voice and an Ftp application for Bluetooth and an Ftp application for WLAN. For each experiment we set d=1 and 3 meters. In addition, in experiment 1, we vary the file size of the Bluetooth FTP application. Each data point collected is averaged over 15 simulation trials using a different random seed for each trial. In addition to the mean value, we verify that statistical variation around the mean values are small and fall within a 95% confidence interval.

A. Effects on Bluetooth Data Traffic

In this experiment, we consider the effects of BIAS and the AFH schemes on the performance of a Bluetooth FTP connection when it is operating in close proximity to a WLAN FTP connection. While the WLAN connection is used to upload a 960 Mbytes file to a server, a Bluetooth FTP connection is used to download files (email, attachment documents) from a PDA to a desktop computer. This latter operation produces similar traffic characteristics than that of a ”HOT SYNC” even if the file sharing protocol used in that case is specific to the PDA manufacturer.

Figure 4(a) gives the packet loss results at the Bluetooth receiver located on the desktop computer. *None* refers to the case when no algorithm is used, while *AFH* and *Scheduling* refer to the use of AFH and BIAS respectively. Also, the distance between

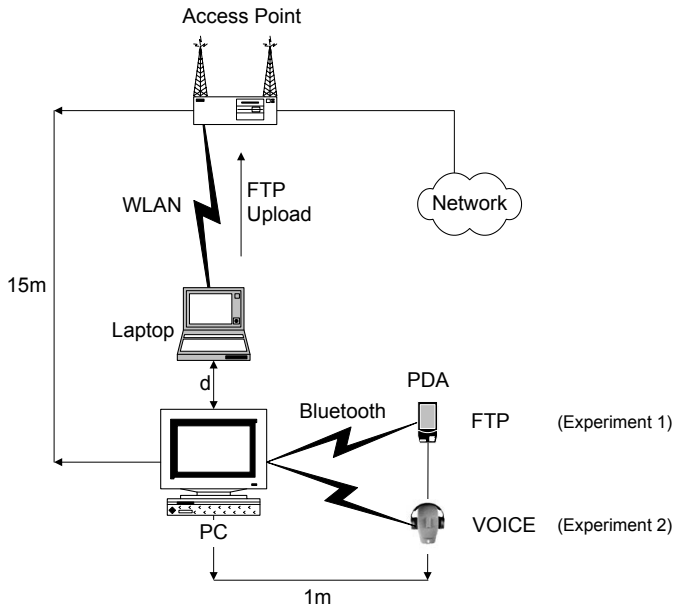

Fig. 3. Topology 1 - Two WLAN devices and one Bluetooth piconet

the Bluetooth desktop and the WLAN laptop is either *1 m* or *3 m* as indicated after the dash. First, observe that the curves are grouped into 3 distinct pairs according to the scheme used. Also, the packet loss corresponding to 1 m is always higher than the one corresponding to 3 m. This is expected since the packet loss is higher when the WLAN node is closer to the Bluetooth device. When no scheme is used the packet loss starts at 12% and 4% for 1 and 3 m respectively. The packet loss for AFH starts at 2% and increases to 6% as the offered load is increased to 800 Kbit/s. There is less than 1% difference between the packet loss for 1 and 3 m. The packet loss for BIAS is negligible and is at least two orders of magnitude lower than the ones observed for *None*.

Note that the relatively higher packet loss observed with AFH depends on the frequency of the synchronization messages exchanged between the Bluetooth master and the slave. There is a trade-off between the communication overhead and the response to changes in the interference environment. A fast responding system will incur a lower packet loss at the cost of a higher communication overhead. In this experiment, synchronization messages are exchanged on average every few seconds (1.5 and 2.5). Since no explicit message exchange is required for the scheduling algorithm, the response time to changes in the interference environment happen within a packet round trip time.

Figures 4(c) and (d) illustrate the TCP goodput and delay results respectively. Observe that the goodput is directly proportional to the offered load until about 480 Kbit/s for all 6 curves. We have computed that about 660 Kbit/s is the maximum application goodput available considering the choice of the simulation parameters. This includes a 10% overhead for the packet headers of all layers between the application and the Bluetooth baseband link and assuming a maximum TCP packet payload of 1460 bytes. Thus, 480 Kbit/s corresponds to 72% of the Bluetooth medium capacity. As the offered load is increased beyond 500 Kbit/s, the difference between the various schemes becomes more significant. The maximum goodput obtained is 600 and 550 Kbit/s with AFH and BIAS respectively. When no algorithm is used the maximum goodput is 480 Kbit/s.

The TCP file transfer delay shown in Figure 4(d) is consistent with the goodput results. The file transfer delay remains below 4 seconds until 500 Kbit/s for AFH and BIAS. It is 2 seconds higher when no algorithm is used. All delay curves take off sharply when the offered load increased above 500 Kbit/s.

In summary, AFH improves the maximum Bluetooth goodput by 25%, while BIAS brings only a 14% improvement. It is important to point out that in this experiment the interference level remains the same for several minutes since the WLAN connection is transmitting during the entire simulation time. Therefore, the throughput advantage brought by AFH can be further increased as the communication overhead is kept low and the channel update interval is increased to several hundred seconds. Had the WLAN traffic been more bursty, additional packet loss could have been incurred with AFH, and the throughput advantage may not have been as significant. On the other hand, BIAS produces a lower packet loss due to its ability to avoid frequencies that have become "bad" within a packet round trip time.

B. Effects on the Bluetooth Voice Application

While in the previous experiment, the objective was to maximize the throughput of an FTP connection, in this experiment the goal is the minimize the delay and most importantly the delay jitter for a Bluetooth voice connection. We use the same parameters used in Experiment 1 and replace the Bluetooth FTP connection with a voice connection as shown in Figure 3. Table II gives the Bluetooth performance results collected on the desktop for d =1 m. The packet loss is 11%, 2.9% and 0.6% with None, AFH

TABLE II
EXPERIMENT 2: BLUETOOTH VOICE PERFORMANCE

	BIAS	AFH	None
d=1 meter			
Probability of Packet Loss	0.0064	0.0294	0.1101
Delay (seconds)	0.0832	0.0014	0.0018
Delay Jitter (seconds)	0.0770	0.0769	0.0767
Goodput (Kbit/s)	2.9096	2.9124	2.9197
d=3 meter			
Probability of Packet Loss	0.0064	0.0155	0.0320
Delay (seconds)	0.0836	0.0015	0.0017
Delay Jitter (seconds)	0.0770	0.0764	0.0768
Goodput (Kbit/s)	2.9109	2.9332	2.9189

and BIAS respectively. Note that the delay jitter is around 76 ms with all three schemes. On the other hand, the delay measured with BIAS is 83 ms, while it is 14 and 18 ms with AFH and None respectively. This result points out the main disadvantage of BIAS in terms of increasing the access delay while lowering the packet loss. However since the delay jitter obtained with BIAS is comparable to what is obtained with AFH and None, then BIAS is still a viable option for voice applications.

The results for d=3 meters are consistent with the discussion presented earlier. In this case the packet loss is lower than with d=1 m since the Bluetooth receiver and the WLAN transmitter are further apart.

C. Effects on the WLAN Performance

Although the interference mitigation schemes presented mostly impact the performance of Bluetooth, it is equally important to consider any effects on the WLAN performance. Before we discuss the effects of the algorithms implemented for Bluetooth on the WLAN, it is important to keep in mind that in the simulation setup used, the WLAN node that is close to the Bluetooth piconet is mainly functioning as a transmitter of data packets and not a receiver. Thus, the impact of the Bluetooth interference is not as significant since the WLAN node only receives short ACK packets. Figure 4(b) shows the WLAN packet loss observed on the WLAN receiver located on the laptop computer. When no interference mitigation algorithm is implemented for Bluetooth, the packet loss is 17% and 10% at a distance of 1 and 3 meters respectively. The packet loss when AFH is implemented drops to 7% and 5% at d=1 and 3 m respectively. The packet loss is less than 1% with BIAS. Note that, we expect the packet loss to be more significant with None and AFH (up to 30% and 15% respectively) when the WLAN node is receiving long packets.

In summary, BIAS not only gives the lowest packet loss results for Bluetooth, but is also a neighbor friendly strategy for WLAN. Since "bad" frequencies can be avoided quickly that reduces the packet loss for both Bluetooth and WLAN.

VI. CONCLUDING REMARKS

In this paper, we study the use of interference mitigation techniques for Bluetooth when operating in close proximity to WLAN systems. We consider a backoff strategy (BIAS) for Bluetooth that avoids the transmission of packets in the WLAN spectrum. We also look at adapting the Bluetooth frequency hopping pattern (AFH) in order to avoid the WLAN spectrum. The former method does not require any changes to the Bluetooth specifications. On the other hand, changing the frequency hopping pattern requires changes to the Bluetooth specifications. The two techniques considered capture the range of solutions considered for the interference problem in the 2.4 GHz band.

Furthermore, while BIAS can be viewed as an intermediate or a temporary fix to the problem, AFH is expected to be part of the next generation Bluetooth specifications and perhaps chipsets if interoperability issues with legacy devices do not hinder its deployment and rapid market acceptance. However, taking a step back from speculative market analysis and technology hypes, our goals in this paper are to examine some of the strategies available for users and vendors and discuss the performance implications and trade-offs they bring.

A summary of our findings is as follows. First, an obvious trade-off lies in terms of communication overhead, and performance improvement. Although partially explored in this study by imposing a synchronization interval, dynamic scenarios where the WLAN interference is intermittent may be difficult to track using AFH. This is probably due to limitations imposed by the communication overhead. The main difficulty is having to dynamically communicate the changes to all slaves in the piconet in order to keep the synchronization. Nevertheless, the use of AFH in environments where the level of interference does not change often, brings additional performance improvements. More specifically, AFH maximizes the throughput for bandwidth hungry applications such as FTP, most file sharing, synchronization applications where the packet loss requirement is not as stringent. On the other hand, the benefits of AFH may not be as obvious for delay jitter and packet loss constrained applications such as voice and video, where packets are never retransmitted and the packet interarrival time is required to be relatively constant. For those applications, BIAS seems to give better performance results, mainly negligible packet loss and low delay jitters.

Finally, our results strongly suggest that no single technique could optimize performance for all scenarios and applications. Perhaps, combining BIAS and AFH could lead to widening the solution space and applying an appropriate technique for each scenario and application considered.

REFERENCES

[1] Bluetooth Special Interest Group, "Specifications of the Bluetooth System, vol. 1, v.1.0B 'Core' and vol. 2 v1.0B 'Profiles'," December 1999.

[2] IEEE Std. 802-11, "IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification ," June 1997.

[3] Federal Communications Commission, "Title 47, Code for Federal Regulations, Part 15," October 1998.

[4] IEEE 802.15.2-2003, "IEEE Recommended Practice for Information Technology -Part 15.2: Coexistence of Wireless Personal Area Networks with Other Wireless Devices Operating in the Unlicensed Frequency Bands," 2003.

[5] J. Lansford, A. Stephens, and R. Nevo, "Wi-Fi (802.11b) and Bluetooth: Enabling Coexistence," in *IEEE Network Magazine*, Sept/Oct. 2001, vol. 15, pp. 20–27.

[6] B. Treister, A. Batra, K.C. Chen, O. Eliezer, "Adapative Frequency Hopping: A Non-Collaborative Coexistence Mechanism," in *IEEE P802.11 Working Group Contribution, IEEE P802.15-01/252r0*, Orlando, FL, May 2001.

[7] N. Golmie, "Bluetooth Dynamic Scheduling and Interference Mitigation," in *to appear in ACM Mobile Network, MONET*, 2004.

[8] Carla F. Chiasserini, and Ramesh R. Rao, " Coexistence mechanisms for interference mitigation between IEEE 802.11 WLANs and bluetooth ," in *Proceedings of INFOCOM 2002*, 2002, pp. 590–598.

[9] N. Golmie, "Bluetooth Adaptive Frequency Hopping and Scheduling," in *Proceedings of MILCOM'03*, Boston, MA, October 2003.

[10] N. Golmie, R.E. Van Dyck, A. Soltanian, A. Tonnerre, and O. Rebala, "Interference Evaluation of Bluetooth and IEEE 802.11b Systems," in *ACM Wireless Network, WINET, Vol. 9, pp. 200-211, May 2003*.

[11] A. Kamerman, "Coexistence between Bluetooth and IEEE 802.11 CCK: Solutions to avoid mutual interference," in *IEEE P802.11 Working Group Contribution, IEEE P802.11-00/162r0*, July 2000.

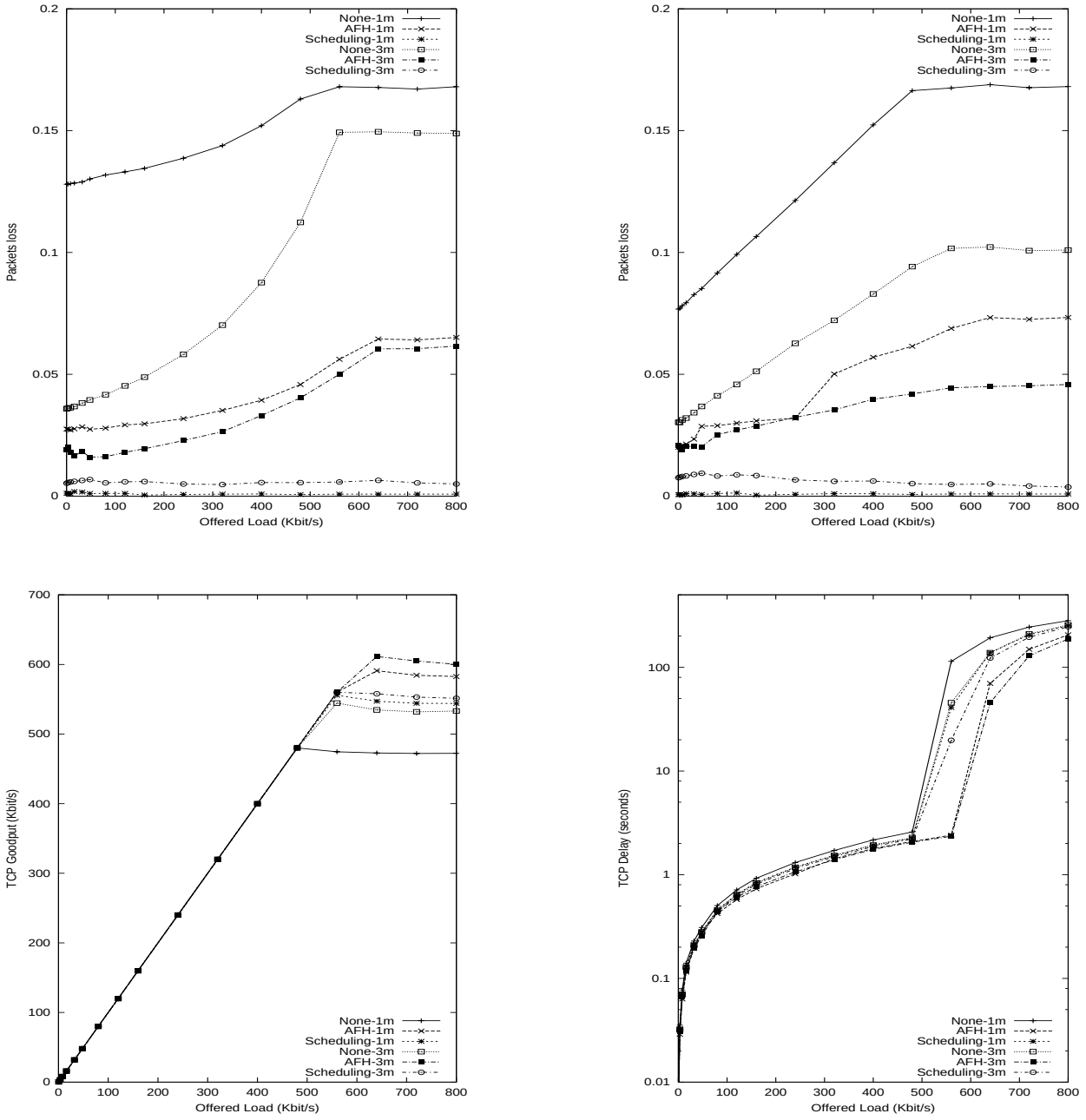

Fig. 4. $\frac{(a)\ (b)}{(c)\ (d)}$ Experiment 1. (a) Bluetooth Probability of Packet Loss (b) WLAN Probability of Packet Loss (c) Bluetooth Goodput (Kbit/s). (d) Bluetooth TCP Delay (seconds)